Graphical CONOPS prototype to demonstrate emerging methods, processes, and tools at ARDEC

Final Technical Report SERC-2012-TR-031

Principal Investigator: Dr. Robert Cloutier, Stevens Institute of Technology

Team Members

Dr. Teresa Zigh, Senior Researcher, Stevens Institute of Technology
Peter Korfiatis, Research Assistant, Stevens Institute of Technology
Behnam Esfahbod, Research Assistant, Stevens Institute of Technology
Peizhu Zhang, Research Assistant, Stevens Institute of Technology
John Santanello, Research Assistant, Stevens Institute of Technology

Contract Number: H98230-08-D-0171 DO4, TTO1, RT0031

Report No. SERC-2012-TR-031
March 23, 2012

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
23 MAR 2012		**00-00-2012 to 00-00-2012**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Graphical CONOPS prototype to demonstrate emerging methods, processes, and tools at ARDEC	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Systems Engineering Research Center,Stevens Institute of Technology ,1 Castle Point on Hudson,Hoboken,NJ,07030**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research has pursued a ?proof of concept? prototype, named the CONOPS Navigator. The Navigator is intended to provide a 3D virtual guide through the development of a CONOPS, and also to integrate various tools and applications currently in use. This integration is a widely-sought capability, one which will enable current CONOPS developers and users the flexibility to import and export analysis parameters and results to and from various familiar and well-used tools. Legacy systems are a fact of life in operational concerns; this prototype is intended to demonstrate interconnectivity on a limited scale between specific simulation and mathematical modeling software packages, via a main operational environment. This environment was built using a game development environment. Although originally conceived as a standalone research task, the potential synergies from merging RT 31 with RT 23, and then combining development architecture and strategies with RT 30, became apparent once development had begun. Some alreadydeveloped external interfaces were seen as adjuncts to the activities performed in RT 30 and these interfaces would certainly be useful in the future. By far, the greater synergies in the development effort were in architecture and operational issues ? such issues are transparent to the user but vital to successful delivery. Further exploration would be toward an integrated data-set and application. The research includes approaches to implementing, managing, and addressing data impedance challenges between applications including Excel, @Risk, and MATLAB. OneSAF was investigated, and the determination is that further training in OneSAF operation is required in order to interface successfully with this tool.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**32**	

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology

ABSTRACT

This research has pursued a "proof of concept" prototype, named the CONOPS Navigator. The Navigator is intended to provide a 3D virtual guide through the development of a CONOPS, and also to integrate various tools and applications currently in use. This integration is a widely-sought capability, one which will enable current CONOPS developers and users the flexibility to import and export analysis parameters and results to and from various familiar and well-used tools. Legacy systems are a fact of life in operational concerns; this prototype is intended to demonstrate interconnectivity on a limited scale between specific simulation and mathematical modeling software packages, via a main operational environment. This environment was built using a game development environment.

Although originally conceived as a standalone research task, the potential synergies from merging RT 31 with RT 23, and then combining development architecture and strategies with RT 30, became apparent once development had begun. Some already-developed external interfaces were seen as adjuncts to the activities performed in RT 30 and these interfaces would certainly be useful in the future. By far, the greater synergies in the development effort were in architecture and operational issues – such issues are transparent to the user but vital to successful delivery. Further exploration would be toward an integrated data-set and application.

The research includes approaches to implementing, managing, and addressing data impedance challenges between applications including Excel, @Risk, and MATLAB. OneSAF was investigated, and the determination is that further training in OneSAF operation is required in order to interface successfully with this tool.

This page intentionally left blank

TABLE OF CONTENTS

Abstract 3
Table of Contents 5
Figures and Tables 6
1 Summary 7
2 Introduction 8
2.1 Use of Gaming Technology as Conduit for Interoperability Communications 8
2.2 Final Platform Selection 10
3 Work Performed 11
3.1 Excel – Interface & Operation 13
3.2 @Risk Simulation 15
3.3 Decision Support Center 17
3.3.1 Vehicle Simulation 17
3.3.2 Vehicle Allocation 21
3.3.3 Response Time 24
4 Research/Questions and Lessons Learned 25
4.1 Research Questions For RT31 25
4.2 Research Lessons Learned 25
4.2.1 Project Management 26
4.2.2 Architecture 27
4.2.3 Project Code/Construction 28
5 Conclusions 30
6 Recommendations for Continuation 31
Appendices 32
Appendix A: references 32

FIGURES AND TABLES

Figure 1 Evaluation of Serious Gaming Technologies 9
Figure 2 ICES Domain 11
Figure 3 ICES External Interfaces 12
Figure 4 CONOPS Lobby 13
Figure 5 Excel Input 13
Figure 6 Excel Output – General Statistics for test data file 14
Figure 7 Browser for storing output as Microsoft Word Document 14
Figure 8 Exported data to Microsoft Word document 15
Figure 9@Risk Simulation - Output of LogNormal Distribution 16
Figure 10 @Risk Simulation - Output of PERT Distribution 16
Figure 11 Vehicle Simulation Initial Screen, MATLAB initialization being performed (JLTV shown) 17
Figure 12 Sample Excel Vehicle Definition File 18
Figure 13 Vehicle Simulation Initial Screen, MATLAB verified (MRAP shown) 3rd Party POV 19
Figure 14 Vehicle Simulation, Overhead POV camera 19
Figure 15 Vehicle Simulation Driver POV 20
Figure 16 Vehicle Capacity Input Screen - Comparing Humvee and JLTV 22
Figure 17 Vehicle Capacity Output 22
Figure 18 Vehicle Fuel Efficiency Initial Screen 23
Figure 19 Vehicle Fuel Efficiency Comparison Output 23
Figure 20 Response Time Input Screen 24
Figure 21 Response Time Output Screen 24

1 SUMMARY

The Department of Defense (DoD) is vigorously pursuing greater efficiency and productivity in defense spending so that it can continue to provide the armed forces with superior capabilities in an environment of flat defense budgets. Toward that end, the Office of the Secretary of Defense (OSD) has issued new acquisition guidance that places increased emphasis on system engineering early in the lifecycle to balance operational performance with affordability and has established the System Engineering Research Center (SERC) to create the tools and processes needed to execute this guidance. As one of its research areas, the SERC has put forth the notion of a concept engineering system for agile CONOPS Development.

Technical Reports SERC-2009-TR-003 and SERC 2010-TR-007 provide a compelling vision, a feasibility assessment, and an initial process definition for Graphical CONOPS development environment for agile systems engineering. Current research will focus on creating an initial prototype to demonstrate a cohesive and easy to use collaborative concept engineering system applicable within the DoD acquisition domain.

Consistent with RDECOM's vision and mission to be the Army's primary source for integrated research, development and engineering capabilities to empower, unburden, and protect the Warfighter, this research topic calls for the creation of an early prototype of the envisioned collaborative concept engineering system demonstrated using RDECOM-ARDEC modeling and simulation infrastructure, RDECOM-ARDEC generated concepts, and RDECOM-ARDEC generated scenarios. Further, it will exercise all three stages of the agile CONOPS development process through the prototype demonstration and assess strengths and weaknesses to guide improvements for future prototypes.

This research pursued the a proof of concept prototype dubbed the "CONOPS Navigator". This prototype provides a 3D virtual guide intended to assist one assigned to CONOPS development, through the setup of individual analysis tools. Further exploration would be toward an integrated set of data modeling tools, able to seamlessly transfer data from one application to another via the CONOPS Navigator main lobby. This task focused solely on the initial stages of data exchange and manipulations between various standalone applications, including Excel, @Risk, and MATLAB. OneSAF was evaluated but not implemented in this task.

2 INTRODUCTION

It is believed that the 3D gaming technologies available today can be used to provide a useful "front end" to the concept engineering process. Selection of the correct game development platform will be critical to this implementation.

2.1 USE OF GAMING TECHNOLOGY AS CONDUIT FOR INTEROPERABILITY COMMUNICATIONS

In early 2011, under RT 003, gaming technology was investigated as the core backbone link between all the CONOPS-specifics functionality – including scenario-building, simulation using various third-party vendor packages, and generating SysML/XML output from vendor offerings already in use by soldiers in the field. To determine which platform to select, a broad range of available gaming environments were examined:

Table 1: Game Development Engines

Torque 2D	Unreal DK	Vicious
Torque 3D	ID Tech (Doom 3)	Open Simulator
Quest 3D	Cry Engineer	C4
Unity	MS-XNA	Gamebryo
Unity Pro	Adobe Flash	Dark Basic
Unreal Engine	Source	Open Simulator

The survey examined qualitative evaluation of each platform on a number of criteria within several overall categories, as shown below:

Features/Capabilities
- Multiplayer
- 3D/2D representations
- Specific comparative strengths and limitations
- Development languages and physics engines supported

Deployment
- Client-Server capability
- Web, PC, Mac supportable
- Minimum CPU and RAM required
- Video card
- Minimum bandwidth

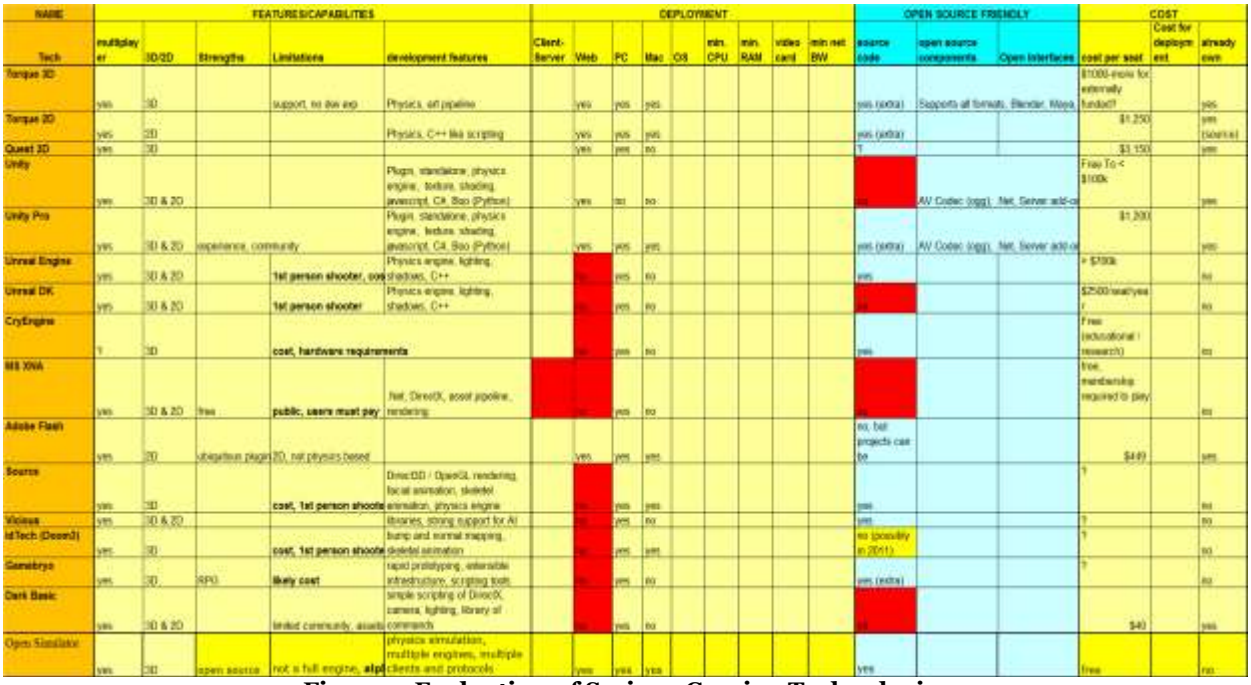
Compatibility with Open Source

- Source code
- Open source components
- Open interfaces

Cost:

- per seat
- to deploy
- license specifications

The evaluations of the software packages/environments along these dimensions are shown in the figure below.



NAME	FEATURES/CAPABILITIES					DEPLOYMENT								OPEN SOURCE FRIENDLY			COST		
Tech	multiplayer	3D/2D	Strengths	Limitations	development features	Client-Server	Web	PC	Mac	OS	min. CPU	min. RAM	video card	min net BW	source code	open source components	Open Interfaces	cost per seat	Cost for deployment	already own
Torque 3D	yes	3D		support, no dev exp	Physics, art pipeline		yes	yes	yes						yes (extra)	Supports all formats, Blender, Maya		$1000-more for externally funded?		yes
Torque 2D	yes	2D			Physics, C++ like scripting		yes	yes	yes						yes (extra)			$1,250		yes (source)
Quest 3D	yes	3D					yes	yes	no						?			$3,150		yes
Unity	yes	3D & 2D			Plugin, standalone, physics engine, texture, shading, javascript, C#, Boo (Python)		yes	no	no						no	AV Codec (ogg), .Net, Server add-on		Free To < $100k		yes
Unity Pro	yes	3D & 2D	experience, community		Plugin, standalone, physics engine, texture, shading, javascript, C#, Boo (Python)		yes	yes	yes						yes (extra)	AV Codec (ogg), .Net, Server add-on		$1,200		yes
Unreal Engine	yes	3D & 2D		1st person shooter, cost	Physics engine, lighting, shadows, C++		no	yes	no						yes			> $100k		no
Unreal DK	yes	3D & 2D		1st person shooter	Physics engine, lighting, shadows, C++		no	yes	no						no			$2500/seat/year		no
CryEngine	?	3D		cost, hardware requirements			no	yes	no						yes			Free (educational / research)		no
MS XNA	yes	3D & 2D	free	public, users must pay	.Net, DirectX, asset pipeline, rendering		no	yes	no						no			free, membership required to play		no
Adobe Flash	yes	2D	vibrant/fun plugin	2D, not physics based			yes	yes	yes						no, but projects can be			$449		yes
Source	yes	3D		cost, 1st person shooter	DirectSD / OpenGL rendering, facial animation, skeletal animation, physics engine		no	yes	yes						yes			?		no
Violeva	yes	3D & 2D			libraries, strong support for AI		no	yes	no						yes			?		no
id Tech (Doom3)	yes	3D		cost, 1st person shooter	bump and normal mapping, skeletal animation		no	yes	yes						no (possibly in 2011)			?		no
Gamebryo	yes	3D	RPG	likely cost	rapid prototyping, extensible infrastructure, scripting tools		no	yes	no						yes (extra)			?		no
Dark Basic	yes	3D & 2D		limited community, assets	simple scripting of DirectX, camera, lighting, library of commands		no	yes	no						no			$40		yes
Open Simulator	yes	3D	open source	not a full engine, alpha	physics simulation, multiple engines, multiple clients and protocols		yes	yes	yes						yes			free		no

Figure 1 Evaluation of Serious Gaming Technologies

Of all the criteria evaluated, several dominated the decision-making process; most of these concerned development and deployment. These included (in no particular order):

- an active and responsive user community,
- the ability to port to different platforms easily,
- the ability to easily support multiple developers,
- providing code control (though this is not a production environment),
- supporting a diversity of programming languages transparently, and

- the ability to either have or incorporate open source components.

In today's environment of flat defense budgets, cost is also a factor, although site-wide and server licenses may help mitigate concerns that per seat licenses may incur.

Although not stated as one of the "critical" components of the decision-making process, the availability of scalable 3D models was also crucial. The applications will be operating in (and as) a visually-based immersive environment; having the models and simulation as realistic as possible will help increase the probability of acceptance and usage by the eventual field users. 3D models can also have a considerable cost factor. For this task, the group utilized 3D models that were found at no cost, although the eventual selected platform does have extensive libraries of 3D models, some available at no cost or for a nominal fee.

Most of the platforms also had other limitations, another factor when selecting the platform – cost and point-of-view being two major considerations.

2.2 FINAL PLATFORM SELECTION

As can be seen in Figure 1, many of the investigated platforms have major drawbacks (shown in red). Chief among these was their inability to deploy on the Web. A secondary consideration for this phase of our research task is the ability of the tool to interface with open source code and components.

The selected platform was Unity 3D Pro. The learning curve for developers was found to be less daunting than that of most of the other platforms, being more intuitive and the facility to develop and deploy components was relatively easily-acquired.

Unity 3D Pro has an asset server which acts as a central code storage and a rudimentary code control mechanism. It has a rich library of models, environments, scripts, and other development components available, either free or at a nominal cost.

Unity 3D Pro supports a number of programming languages: C#, Boo (Python), and Javascript. The Unity physics engine supports movement, collision, gravity for solid objects, and users can modify textures/meshes. This ability will be critical if terrain generation from various USGS databases is to be evaluated.

Unity 3D Pro has a large user community which is extremely responsive to posted questions, and a forum containing posted solutions to many commonly-found problems or desired effects. As this research task was focused mainly on interfaces between 3rd-party software, we did not find solutions in user community resources for these tasks, however the resources did help when implementing some of the more complex model representations and movement.

3 WORK PERFORMED

Kickoff	1/24/2011
In Progress Review (IPR) 1	5/12/2011
In Progress Review (IPR) 2	9/12/2011
In Progress Review (IPR) 3	11/28/2011
Final Review/Presentation	3/23/2012

This work was performed in two stages:

- The first stage was the development of standalone applications, to validate the conceptual and architectural approach of each interface separately.
- The second was to merge the various standalones into one umbrella executable.

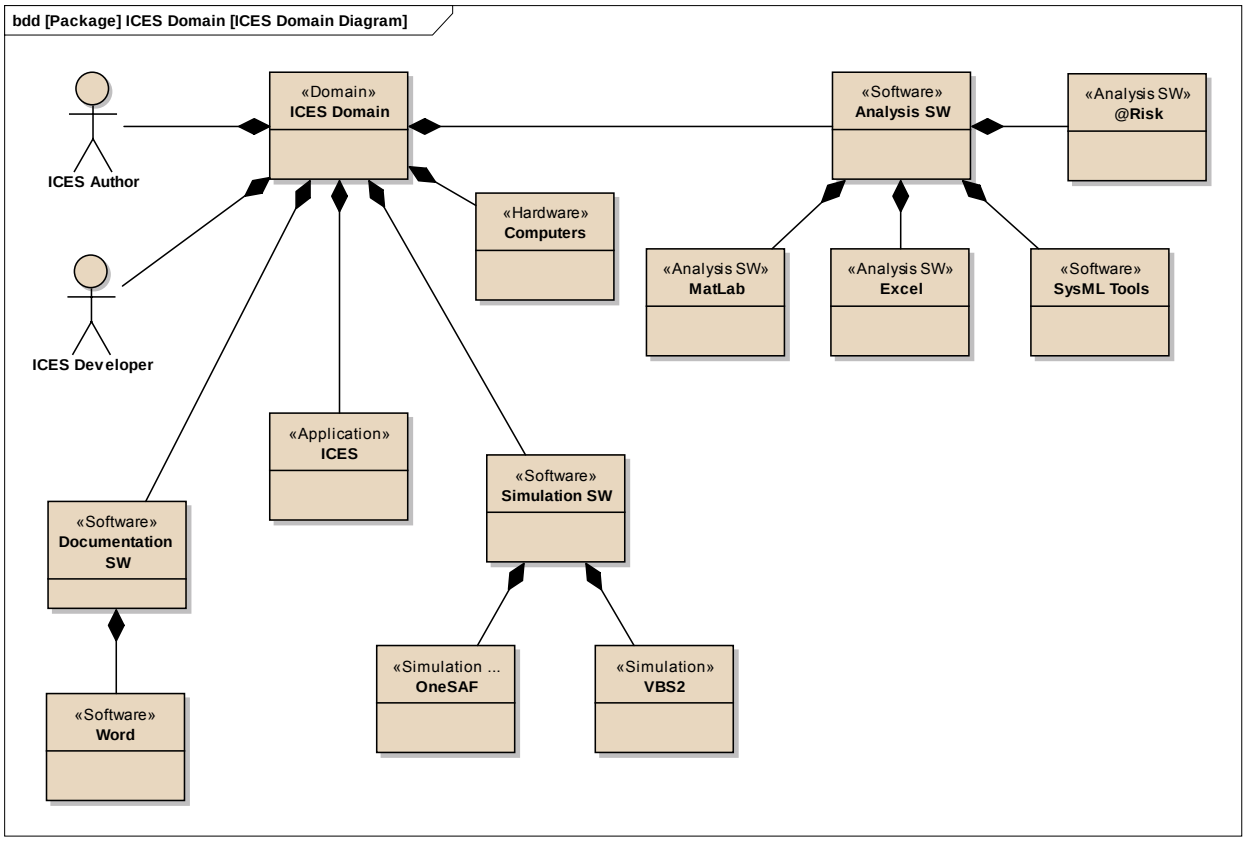


Figure 2 ICES Domain

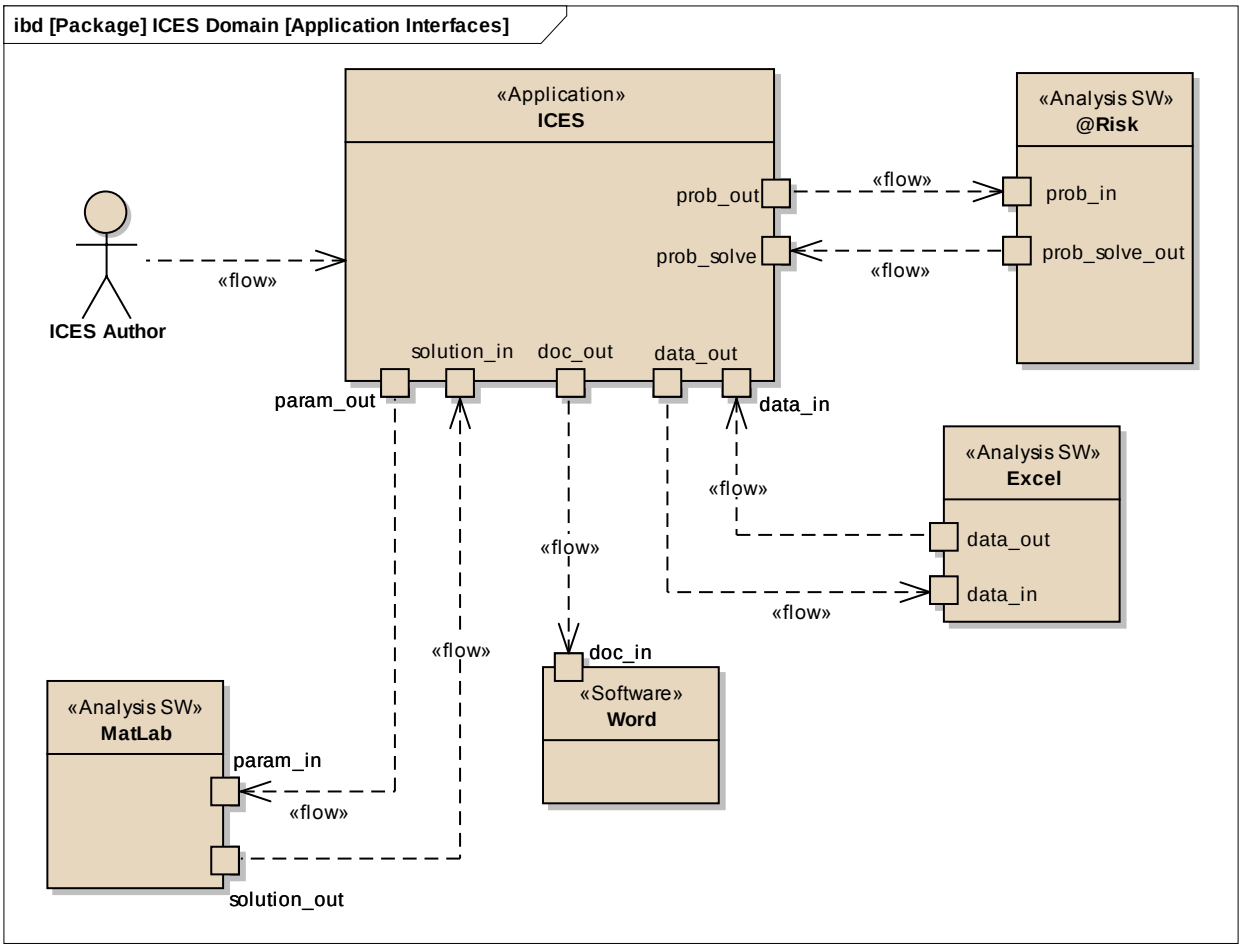


Figure 3 ICES External Interfaces

This is a proof-of-concept research task – that implies that the software must perform within a relatively flexible set of criteria; it is not a production system. Of necessity, major error-handling is not a factor in our evaluation of preparedness, but reasonable error-handling and performance issues are addressed.

Our first step was to develop the interface between Unity and Excel – and this approach brought us an unintended benefit. An interface for Excel would also be usable for Microsoft Word, therefore creating a conduit to save results from simulations run in third-party software. We began with a CONOPS Lobby – a virtual room where a use could choose among several options for their particular need.

- Microsoft Excel
- Anylogic
- @Risk Simulation libraries
- OneSAF
- Sparx (SysML package)
- MATLAB (via the Decision Support Center, DSC)

The opening scene is shown below:

Figure 4 CONOPS Lobby

3.1 EXCEL – INTERFACE & OPERATION

Upon the selection of Excel, the following right- and left-hand side menus appear:

Figure 5 Excel Input

The software allows the user to specify a data file for input. Once entered, as shown below, the user can select from various result options. Here, we show the output

resulting from the selection of all the available general statistics for the dataset provided in the test file:

Figure 6 Excel Output – General Statistics for test data file

The user is then given the option to export the results data directly to a file which can be stored, or to open the results data in a Microsoft Word document, for further viewing or possible manipulation.

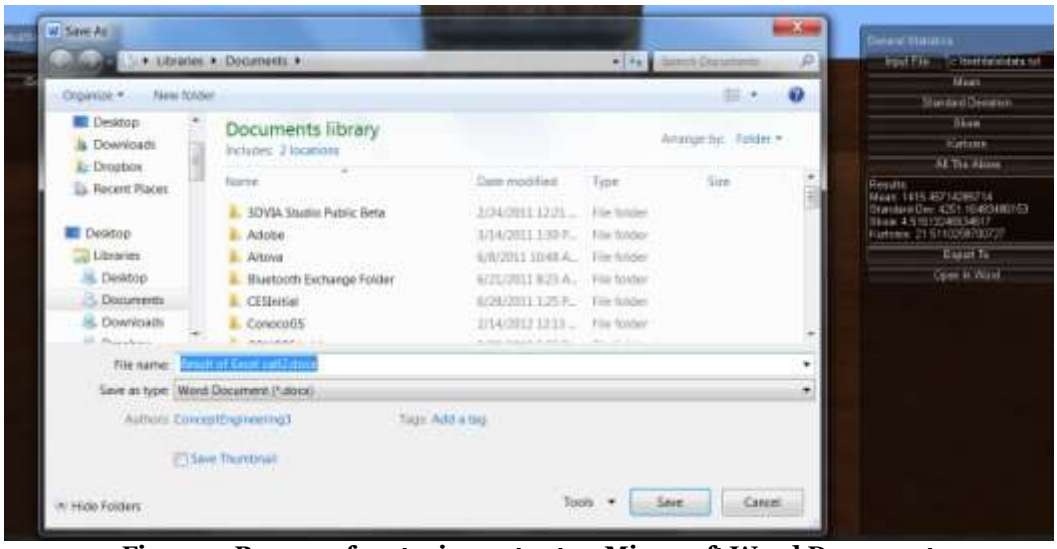


Figure 7 Browser for storing output as Microsoft Word Document

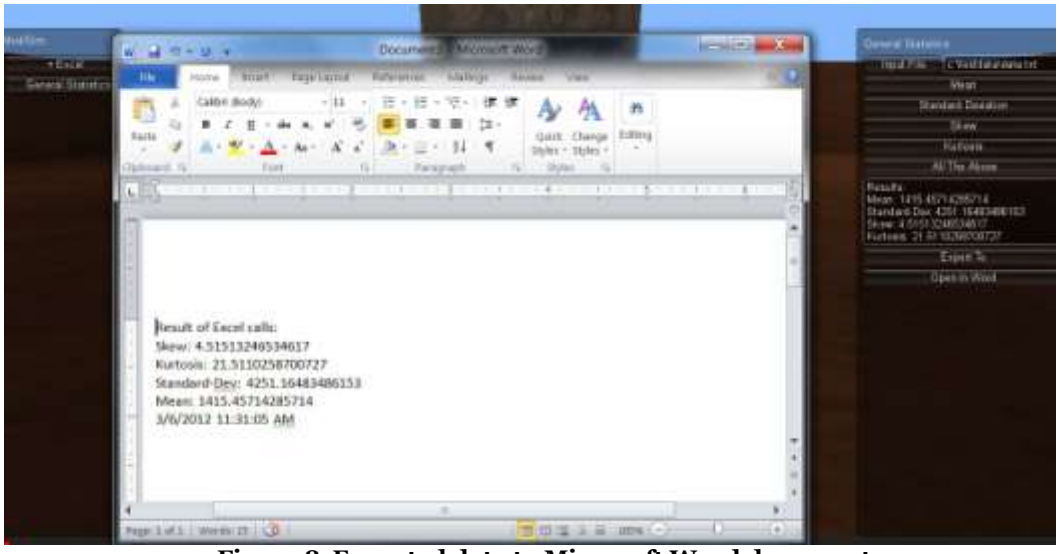


Figure 8 Exported data to Microsoft Word document

The use of Excel is enabled by C# scripts within Unity 3D Pro, and uses two external programs for initiating IO Pipes. The two external programs reside in a special Deploy folder, and must be present for the application to successfully call the Microsoft Excel functions, as well as writing to a Microsoft Word document. This is an example of the synergy of this development, as well as the benefits of using named pipes. A named pipe is an extension of the pipe concept on Unix-type systems, and serves as the inter-process communication conduit for the data stream input and output. A named pipe is system-persistent and exists beyond the life of the process, which requires that it be deleted once it in no longer needed. Once the process connects to the named pipes, communication between applications is possible.

3.2 @RISK SIMULATION

The selection of the @Risk Simulation libraries leads to similar input screens, although they are tailored for individual input - characteristics of the distributions which serve as input to the libraries.

Figure 9@Risk Simulation - Output of LogNormal Distribution

Figure 10 @Risk Simulation - Output of PERT Distribution

The calls to the @Risk simulation SDK libraries are made via Javascript. The return values are text, and the graphic representation is also formatted as a stream of text.

3.3 DECISION SUPPORT CENTER

The decision support application is partitioned into three sections, each of which highlights a separate interface.

3.3.1 VEHICLE SIMULATION

Upon selection of the Decision Support Center application, Vehicle Simulation, the following initialization screen is displayed.

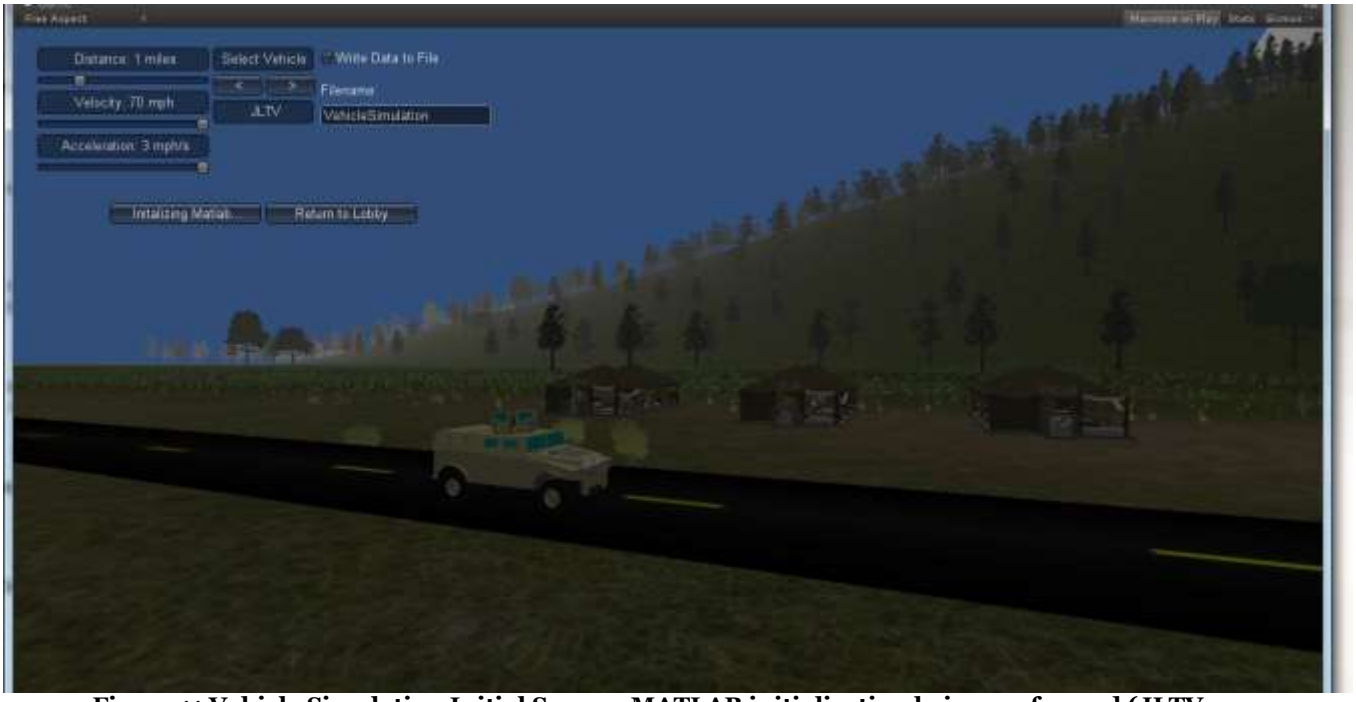


Figure 11 Vehicle Simulation Initial Screen, MATLAB initialization being performed (JLTV shown)

The user can use the slider bars shown in the above figure, to vary the distance of the simulation, the speed and acceleration of the vehicle. The application retrieves vehicle specifications and parameters from an Excel file. In this file, each sheet represents the specifications of a vehicle – the file can be extended and modified as necessary for additional vehicles (see Figure 12 below).

Vehicle 1			
Name:	Humvee		
Vehicle Attributes			
Speed Distribution	RiskLognorm		
Speed (mean)	46		
Speed (std dev)	12		
MPG	20		
Personnel Capacity	2		
Max Speed	65		
Max Acceleration			
Capacity model			
Internal Variables		*External Variables*	
Personnel Capacity	2	Required Passenge	53
MPG	20	Distance	125
		Cost of Fuel	3.3
Calculations			
Trips Required	27	Trips	
Total Distance to Travel	6750	Miles	
Fuel Use	337.5	Gallons	
Fuel Cost	1113.75	Dollard	
Travel Time (average)	146.7391304	hours	
Fuel Economy Model			
Internal Variables		*External Variables*	
MPG	20	Distance	125
		Cost of Fuel	3.3
Calculations			
Fuel Use	6.25	Gallons	
Fuel Cost	20.625	Dollars	
Response Time Simulation			
Internal Variables		*External Variables*	
Distribution Type	RiskLognorm	Distance	125
Speed mean	46	Cost of Fuel	3.3
Speed SD	12		
MPG	20		
Calculations			
Average Speed	60	MPH	
Response Time	2.083333333	Hours	
Fuel Use	6.25	Gallons	
Fuel Cost	20.625	Dollars	

Figure 12 Sample Excel Vehicle Definition File

The application also shows an initialization of MATLAB, prior to running the application. If MATLAB is not installed, the user will not be able to run the simulation. The initialized application is shown in the next three figures; the first is a 3rd-person view, the second is the overhead point of view built into the application, and the third is a 1st-person "driver" view from the vehicle interior.

Figure 13 Vehicle Simulation Initial Screen, MATLAB verified (MRAP shown) 3rd Party POV

Figure 14 Vehicle Simulation, Overhead POV camera

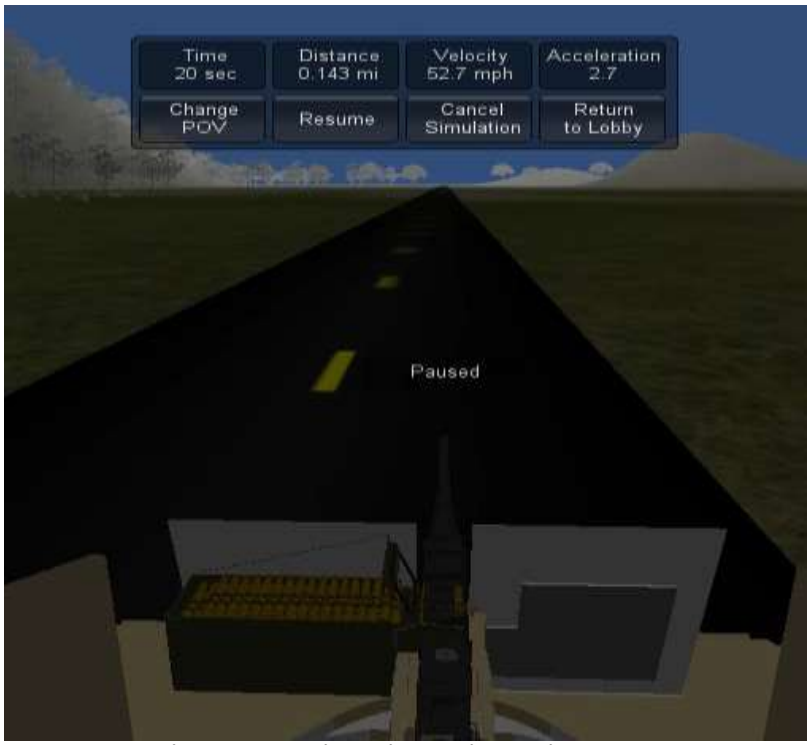


Figure 15 Vehicle Simulation Driver POV

The algorithm to model ideal one-dimensional motion of a vehicle over a specified distance assuming a maximum velocity, acceleration, and jerk used in this simulation is based loosely on the work of Richard D Peters (Peters). This algorithm runs iteratively calculating the parameters to model the vehicle at each step of time and accounts for the four possible outcomes of motion:

- max velocity is reached,
- max acceleration is reached but not max velocity,
- neither max velocity nor max acceleration is reached, and
- max acceleration is not reached but max velocity is reached.

The MATLAB program was then integrated with the Unity platform to show a real time representation of this data in a visual simulation. Unity creates a TCP/IP listening server, opens MATLAB, connects as a client to the Unity application on the specified port, sends a request for data, and then waits. During this time, the user on the Unity application is given time to choose a vehicle, distance, max velocity, and max acceleration. Once MATLAB initializes, the user is then given the option to run the simulation. As the simulation button is pressed, data is passed through the TCP/IP connection to MATLAB which interprets the input and begins running the simulation. On each iteration MATLAB first checks for a command from Unity, then calculates the next set of parameters, and sends them to Unity over the network connection. As Unity

gets the data packets, it converts them to the distance velocity and acceleration arguments, moves the vehicle appropriately on the next frame and updates the display for current position, velocity, and acceleration. At any point during the simulation, the user can pause the simulation, restart the simulation with the same or different parameters, or cancel the simulation and exit to the main menu. This is achieved by sending a command packet to MATLAB over the established TCP/IP connection and allowing the MATLAB program to process the command and act accordingly.

The current basic formulation of the MATLAB model does not yield overly powerful results, but it proves the concept of a real time simulation built around the computational power of MATLAB and the visual properties of Unity.

Future simulations could include more powerful formulations and one investigation can include a feedback loop from Unity. For example, a more complete model could be created for the vehicles, including properties like torque and mass. A 3D path could be created in Unity for the vehicle to follow and, as the vehicle moves along that path, data could be sent to MATLAB concerning the pitch and yaw of the vehicle, which would affect its velocity and acceleration characteristics. As this data is sent to MATLAB in each frame, the subsequent calculation would be sent back showing new displacement acceleration and velocity in each direction as well as about each axis.

3.3.2 VEHICLE ALLOCATION

Upon selection of the Decision Support Center application, Vehicle Allocation, the user can select the comparison of vehicles for various parameters, the first one shown below, is vehicle carrying capacity – in this case, between a Humvee and a JLTV.

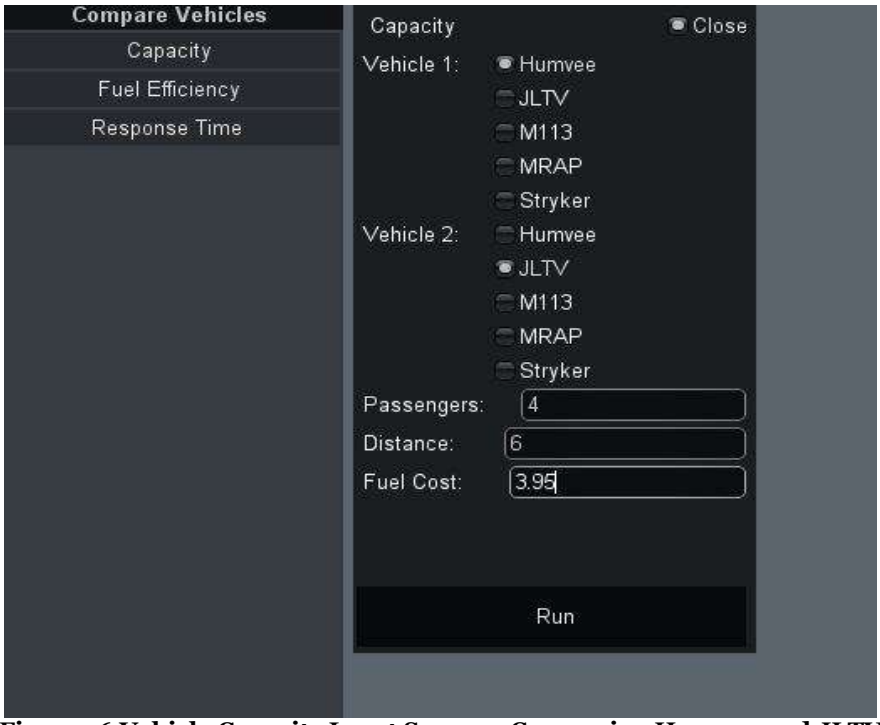


Figure 16 Vehicle Capacity Input Screen - Comparing Humvee and JLTV

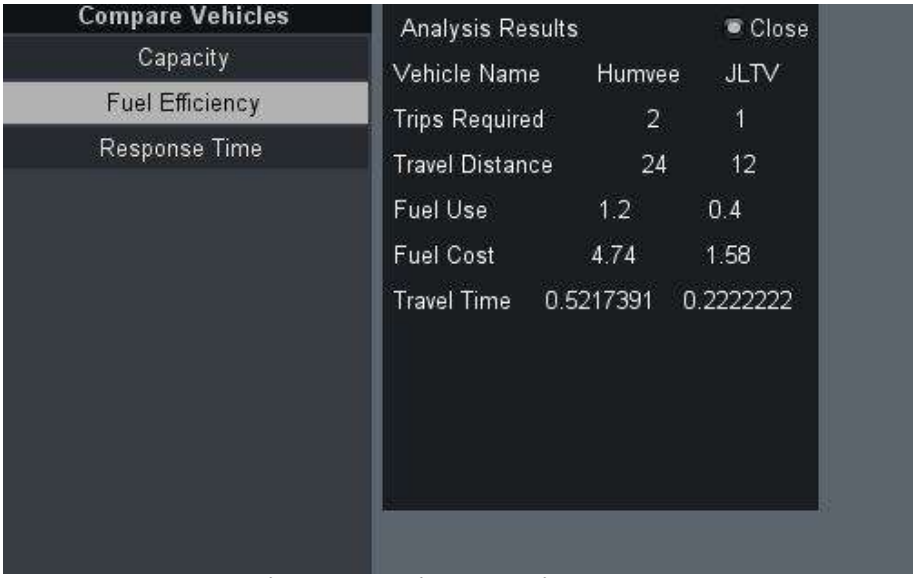


Figure 17 Vehicle Capacity Output

In order to run the vehicle fuel efficiency calculations, the initial screen presented is:

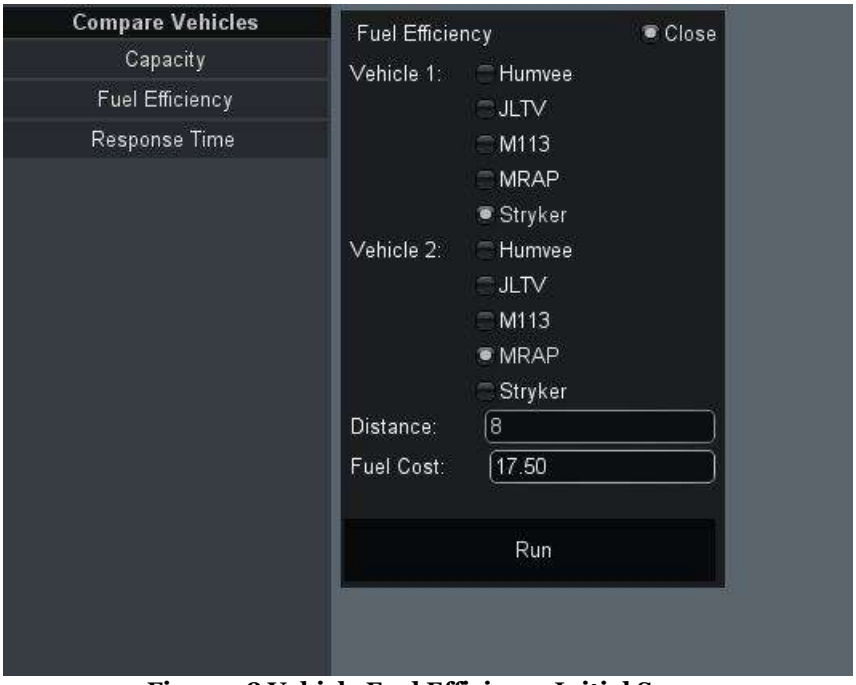


Figure 18 Vehicle Fuel Efficiency Initial Screen

In this case, the vehicles being compared are a Stryker and an MRAP, over a distance of 8 miles and with a fuel cost of $17.50/gallon. The output from this simulation is shown below:

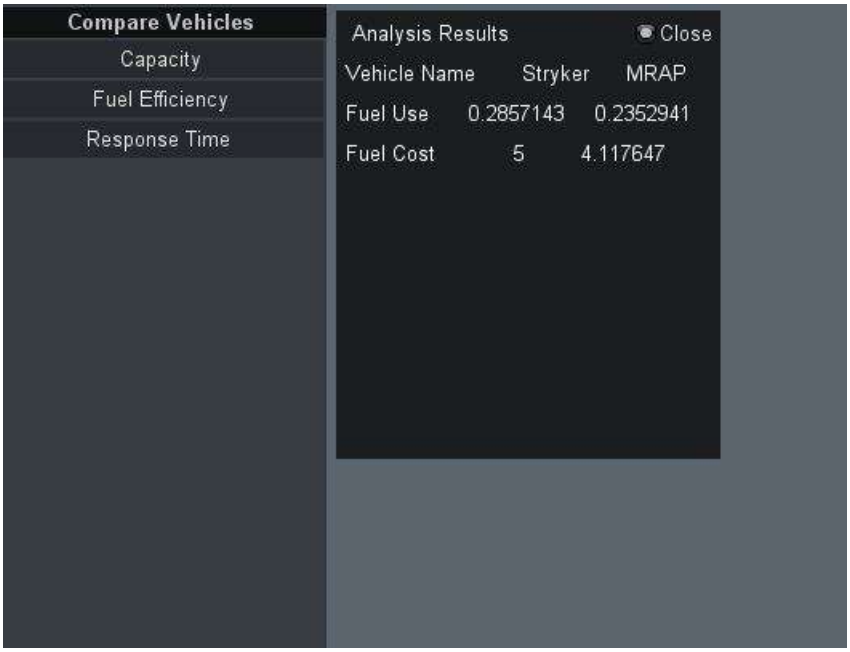


Figure 19 Vehicle Fuel Efficiency Comparison Output

3.3.3 RESPONSE TIME

Upon selection of the Decision Support Center application, Response Time, the user can compare the fuel usage and fuel cost between two vehicles traveling the same distance.

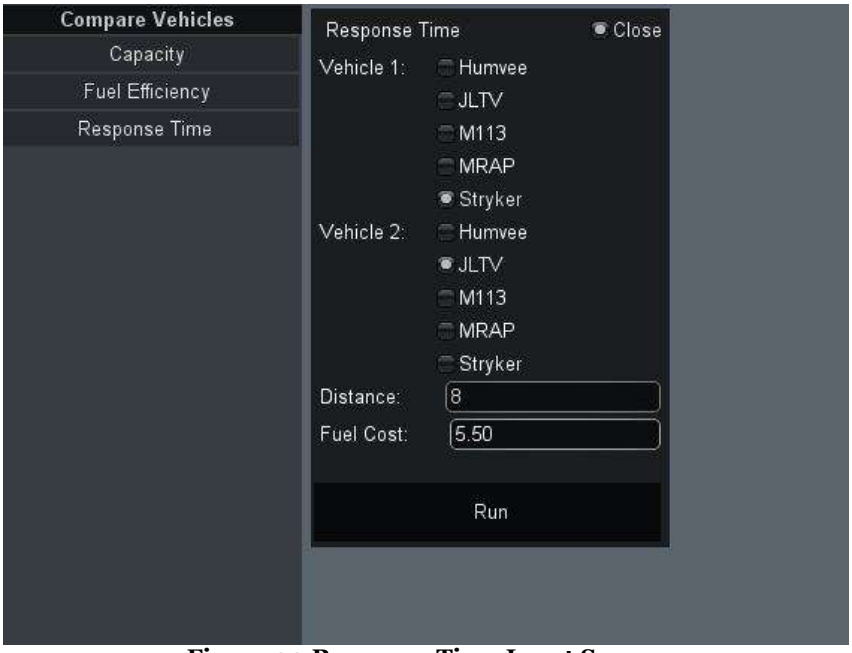


Figure 20 Response Time Input Screen

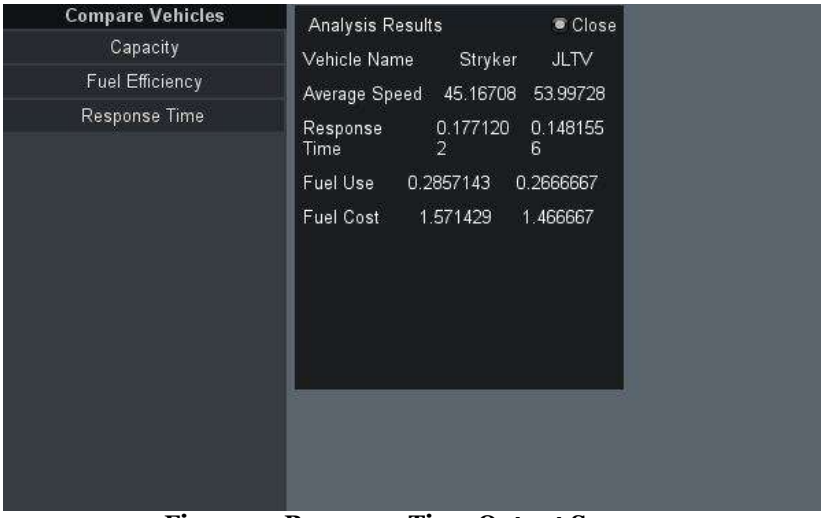


Figure 21 Response Time Output Screen

The calculations for the vehicle comparison Capacity and Fuel Efficiency decision components are being made via the Excel interface. The Response Time simulation is handled using the @Risk Simulation SDK library function.

4 RESEARCH/QUESTIONS AND LESSONS LEARNED

In addition to RT 30, RT31 evolved from the initial research task, RT3. Since RT31 seeks to address related sponsor defined needs, the research team defined a high level research question to tie together the RT 3/30/31 thread:

Can the process of Concept Engineering improve the understanding and development of a concept of operations using gaming technologies along with an interactive, collaborative, and graphical environment?

From this question, each research task contains lower level research questions to address specific task goals.

4.1 RESEARCH QUESTIONS FOR RT31

- Can the process of CONOPS modeling and simulation be improved through the use of a graphical user interface which would serve as a conduit for data?
- What are the benefits of a single user interface for the tools currently in use for modeling and simulation studies?
- What are the drawbacks of a single user interface for the tools currently in use for modeling and simulation studies?
- Does real-time collaboration between distributed stakeholders improve the CONOPS development in the area of modeling and simulation?
- Can a real-time collaboration environment enable quicker consensus on CONOPS generation?
- Are there new or specific issues in asynchronous software development in an immersive environment?

4.2 RESEARCH LESSONS LEARNED

The software effort is actually secondary to the research questions being posed, yet consumes what seems to be 98% of effort in the early stages. Several categories of "lessons learned" were observed during this effort:

4.2.1 PROJECT MANAGEMENT

- It is critical to continuously monitor migration to new development environment releases – we now only migrate as a team, and then only after testing current builds in new release.

- Iterative development tasks can lead to redundant efforts and conflicts; a lot of time and effort are needed to avoid wasted effort or design conflict.

- Use of Agile processes very difficult in academia – neither students nor faculty is regular in their schedules/work times. Because of this, the use of Skype and Google+ hangouts can be effective, especially when it comes to review, walk-through, and testing.

- Weekly builds, although difficult to implement at the beginning of the architecting phase, are a critical factor for successful completion. The reasons are two-fold:
    - They also build team cohesiveness, and focus attention on research and project goals
    - They highlight any performance or operational anomalies – code that works well in the Unity development environment, may not be operational in executable build. This could give a false sense of security to developers, by effectively hiding executable conflicts.

- International composition of workforce has its own challenges, both from a language side and a cultural side. In addition, citizenship requirement for some software was a mitigating factor.
    - Clear Skype or Google+ hangouts, or even written word communication can be challenging and clarity can suffer when there's a lack of visual clues
    - Video conferencing is highly preferred over voice-only or written communication.
    - Face-to-face remains the best way to manage, but video is a valuable 2nd best
    - Avoid idioms when describing operational desired design attributes and functionality
    - Analogies can work, but should be simple and clear

- Measuring progress via visible functionality is not helpful, nor is using long-standing measures such as SLOC. Other criteria must be adopted and we propose a combination of SLOC count and a partitioning of categories of code: infrastructure, actual 3D object presentation, and 3D model manipulation
    - Current Statistics:
        - SLOC for executable: 1270, # objects: 147
        - SLOC for work in progress: 416, # objects: 63

- Organization of code within the project listing is critical, especially when using a multiple-developer approach. Naming folders clearly and grouping scripts together is critical, since most of the scripts are small (and again, naming clearly here is critical to efficient searches).

- The actual 3-D models used are free for academic and research use but are not free for commercial distribution; this is a consideration if deploying in an organization, so factor time for 3-D model development.

- The OneSAF battle simulation package is highly complex and requires a great deal of time and training to generate output capable of being used by the current application.

4.2.2 ARCHITECTURE

- Integration of previously-developed standalones into an overall executable build was moderately difficult. After consideration, it was felt that clear architecture and preview of all modular activity and interfaces would be a more practical approach.

- Early integration (and builds) would also assure that the look-and-feel of various tool interfaces is similar.

- All architecture and data objects should be specified as completely as possible and as early as possible.

- Because architecture (and infrastructure) is not "seen," the work done is not obvious to management/ customer – and therefore it gives the impression of "no progress"

- Error handling architecture should be context-reliant, and needs to be addressed for consistency

- Communication issues were the largest stumbling block in this particular research task. The interfaces between each tool and the Unity 3D gaming platform are all custom built, but also built as generically as possible, to support a wide range of calling options.

- One of the major re-writes/re-working of architecture centered on the performance of the communication interfaces between the Unity platform and MATLAB and the @Risk libraries. Although this is "proof of concept" software, it was felt that some performance considerations needed to be taken into account, if the software is to be successfully deployed on a large scale and over multiple platforms with multiple users.

4.2.3 PROJECT CODE/CONSTRUCTION

- Asset Server Change Control/Staging Platform
    - Individual project access in asset server needs to be transparent to all developers
    - Using the control management Asset Server was more complex than anticipated, and there was no easy way to roll-back to a previous release or version
    - There were occasional slow-downs when committing to or updating from the Asset Server

- Highly-modular design vies with programming strategies – optimal breakpoints must be developed
    - Assignment of modular design elements can be problematic and, because of the iterative nature of design and development, is a real challenge

    - Realize that graphic design of 3D models, including scaling and manipulation, took longer than originally anticipated; this is not due to the provenance of the models, but is inherent to 3D environments

- Avoid manipulation of the object surface meshes – in order to indicate a "selection" insert an indicator above the object itself

- When working with animation and interfaces with simulation software, performance and communication issues may take additional development time.

- Actual physical movement of groups (for instance, a group of ground vehicles, or deployment of fleet) will be crucial as the design moves forward, in terms of visual representation, as well as generated output

- Although containment of objects isn't a large consideration in this RT yet, it should be kept in mind as an issue which can impact performance, storage, and retrieval.

- Manipulating colliders is how objects have solidity in the environment - this is an important consideration when making scenes executable.

- Consider the use of multiple cameras as a default mechanism for each scene when being built, and provide an easy toggle for users to switch views.

- Terrain generation from real-world (USGS) server files is inefficient at this time. The size and complexity of the USGS library files were too large to scale effectively in the task time frame. In addition, the problem of ground cover (tree canopy, etc.) is an issue when the aim of the simulation is to depict actual ground condition. The team was able to successfully import small ground segments via Google Earth and Google SketchUp from areas which has little or no ground cover.

5 CONCLUSIONS

The possibility of using a 3D gaming environment and platform as a window into the interfacing and use of various simulation packages is proven. Three third-party packages (MATLAB, Excel, @Risk) were integrated within the Unity 3D gaming environment, with a bookcase paradigm for simulation package selection.

Licenses for third party software must be investigated for wide-spread distribution for this software – the development group has used academic licenses to implement the project with the accompanying software, but production deployment will necessitate a close examination of distribution and use of independent vendor libraries and packages.

6 RECOMMENDATIONS FOR CONTINUATION

The strategies for continuation include the following:

Future work would encompass:

- the investigation of interfaces with AnyLogic, Sparx, and OneSAF
- terrain-mapping capability

With regards to deployment, future work would include the investigation of multiple platform operations, as well as the ability for multi-users to both operate and co-operate within the tool.

Multiple military vehicle types should be considered, as well as movement in three dimensions – this could include land, water, and air-based vehicles. Along with terrain-mapping, the application of uneven surfaces, and the consideration of pitch, rolls, and yaw inherent in 3D motion would significantly enhance the capabilities of the CONOPS Navigator.

We intend to develop analytical metrics which incorporate both commonly-used code metrics and the complexity of interaction between the interfaces and Unity 3D Pro scenarios and 3rd party software.

Another proposed avenue of investigation is to utilize Magic Draw's animation capability and apply it to the CONOPS Navigator architecture evolution.

APPENDICES

APPENDIX A: REFERENCES

Peters, R. D. (n.d.). *Ideal Lift Kinematics - Complete Equations for Plotting Optimum Motion.* Retrieved January 2012, from www.peters-research.com: http://www.peters-research.com/index.php?option=com_content&view=article&id=61%3Aideal-lift-kinematics&catid=3%3Apapers&Itemid=1